Alignment of ReMMo with RBAC to Manage Access Rights in the Frame of Enterprise Architecture

Feltus, Christophe; Petit, Michaël; Dubois, Éric

Published in:
2015 IEEE 9th International Conference on Research Challenges in Information Science (RCIS)

DOI:
10.1109/RCIS.2015.7128887

Publication date:
2015

Document Version
Early version, also known as pre-print

Alignment of ReMMo with RBAC to Manage Access Rights in the Frame of Enterprise Architecture

Christophe Feltus, Eric Dubois
Luxembourg Institute of Science and Technology
5, avenue des Hauts-Fourneaux, L-4362 Esch-sur-Alzette, Luxembourg
name.firstname@list.lu

Michaël Petit
Faculty of Computer Science,
University of Namur, Belgium
mpe@info.fundp.ac.be

***Abstract*—Aligning the business operations with the appropriate IT infrastructure is a challenging and critical activity. Without efficient business/IT alignment, the companies face the risk not to be able to deliver their business services satisfactorily and that their image is seriously altered and jeopardized. Among the many challenges of business/IT alignment is the access rights management which should be conducted considering the rising governance needs, such as taking into account the business actors' responsibility. Unfortunately, in this domain, we have observed that no solution, model and method, fully considers and integrates the new needs yet. Therefore, the paper proposes firstly to define an expressive Responsibility metamodel, named ReMMo, which allows representing the existing responsibilities at the business layer and, thereby, allows engineering the access rights required to perform these responsibilities, at the application layer. Secondly, the Responsibility metamodel has been integrated with ArchiMate® to enhance its usability and benefits from the enterprise architecture formalism. Finally, a method has been proposed to define the access rights more accurately, considering the alignment of ReMMo and RBAC. The research was realized following a design science and action design based research method and the results have been evaluated through an extended case study at the Hospital Center in Luxembourg.**

Keywords-component; Access rights, Business/IT alignment, Responsibility model, Enterprise architecture, ArchiMate, RBAC.

1. INTRODUCTION

In the current complex and evolving environment, aligning the business down to the appropriate IT infrastructure is a challenging activity that needs to be carefully handled. One aspect of this alignment concerns the access to data and applications required by employees depending on the information they need to perform business activities. In this area, our review of the access rights models and engineering methods [1] has highlighted an evolution towards more consideration of business concepts. Indeed, the access rights management solutions gradually progresses towards a wider integration of the business concepts such as employees' obligations and responsibilities regarding the tasks they are assigned to.

In parallel, many governance standards and norms have acknowledged the above alignment challenges and have highlighted new needs to be satisfied in terms of business/IT alignment and access rights management. The review of these standards and norms [1] has allowed depicting concepts to be taken into account when addressing governance requirements. Unfortunately, we have observed that the access rights management solutions do not yet fully consider and integrate these concepts. For the moment, there exists no solution allowing thoroughly connecting the business layer and application layers and, alike, no common model is yet agreed upon among business and the IT staff, especially concerning the management of the access rights. Despite our observation related to the need for considering the concept of responsibility, as well as a set of concepts that allow defining it, such as the accountability, the capability or the right to use, to date no approach really considers these concepts. Hence, this observation has led us to analyze the literature from the field of IS/IT and from the field of the human sciences. The literature analyzed has allowed us to discover the breathiness of the notion of responsibility that gathers, at the same time, information related to (1) the accountabilities of the employees, which are mainly defined at the business layer. (2) the rights and capabilities that these employees require to perform their accountabilities. These rights and capabilities are issued from the business layer but impact the application layer, with e.g. the definition of the access rights. And (3) the assignment of responsibilities to employees, directly or through the roles they play. Knowing the meaning and acknowledging the importance of these concepts led us to the definition of the Responsibility metamodel (named ReMMo) that allows defining the responsibilities at the business layer and, thereby, allows engineering the access rights required to perform the responsibilities, to be provisioned at the application layer.

To enhance the usability of ReMMo, we have exploited an enterprise architecture (EA) model. EA consist in approaches which enable illustrating the inter relations between the different layers of a company and between the different aspects that it addresses such as the behavior, the information, or the people. EA metamodels provide views which are understandable by all the stakeholders and which allow making decisions, knowing the impact over the company. However, the problem with the EA metamodels is that, in general, the concepts which compose them lack precision and, therefore, are hardly usable to perform, verify or justify concrete alignments. Acknowledging this statement, we concluded that, in practice, EA metamodels do not permit accurate engineering of the access rights to be provisioned to the employees at the application layer, based on the specification from the business layer such as required by the governance standards and norms. In parallel, we also considered that the EA metamodels provide a good basis for this since they model the most significant concepts related to the information system of a company. To reap the benefits of the enterprise architecture metamodel for the engineering and the management of the access rights, we have decided to focus our research

on integrating the Responsibility metamodel with the business layer of the ArchiMate EA metamodel.

Using the concept of role for the management of access rights is an approach commonly agreed upon by most of the companies. RBAC is the leading model in this area and is based on two processes: the assignment of users to roles and the assignment of permissions to roles. RBAC is a model that allows optimizing, at the application layer, the assignment of a large number of permissions to a large number of roles. Throughout the literature, we have accordingly observed that the RBAC role is mostly considered as a business role. In practice, however, we see that the concept of role is used at the application layer, where application roles are exploited. This was, for instance, the case in our case study at the hospital. The non-alignment between the business roles and the RBAC roles led us first to align our Responsibility metamodel with RBAC. This alignment has allowed the tracing of the relationships, amongst others, between the user and the employee, between the RBAC role and the business role, and between the permission and the responsibility. Secondly, we have proposed an Access rights management reference model, at the business layer of ArchiMate. This reference model has been designed in a way that it may be supported, at the application layer, by RBAC based solution, and hence, by the Band's reference model. In practice, the alignment of ArchiMate extended with the Responsibility metamodel with RBAC, and the processes modelled in the Access rights management reference model, constitutes our method for the engineering of access rights management based on the employees' responsibility. In this method, the concept of responsibility is used as a pivot between the business layer and the application layer. It offers the advantage to integrate the requirements from both layers, namely: on the first hand, at the business layer, employees are gathered in business roles and those business roles are organized in an organizational chart and are assigned to a set of responsibilities which concern a precise business task; on the other hand, at the application layer, the assignment of access rights to the employees is optimized using the concept of RBAC role which gathers all the permissions required by a business role.

In this paper, we first present the Responsibility metamodel in Section II.A and we propose a language for expressing the responsibilities in the frame of enterprise architecture in Section II.B. Then, we map ReMMo and RBAC in Section III, and, based on this mapping, we provide a model (is named the *Access rights management reference model)* to sustain the management of the access rights based on the employees' responsibility in Section IV. A real case study at the Hospital Center in Luxembourg is presented in Section V. The evaluation criteria of this case study is the enhancement of the accuracy of the access rights. Related works are introduced in Section VI and Section VII concludes the paper and provides some future works.

II. REMMO

This section presents first the Responsibility metamodel UML diagram and second the language which support the usability of this model.

A. ReMMo modeling

To model the responsibility, we analysed in details what the concept of responsibility means and how it allows connecting the access rights and the business roles. Enhancing the understanding and the modelling of the responsibility contributes, on the first hand, to improve the definition of the role played by the employees and, thereby, the management of the access rights they need, and on the second hand, to satisfy the governance needs corresponding to the responsibility.

We propose ReMMo, a Responsibility metamodel for modelling a rich concept of responsibility, the accountabilities that are part of it and its links with the employees, the business roles, the tasks, and the rights and capabilities. Figure 1 represents, in UML, the main concepts of the Responsibility metamodel. The classes in yellow correspond to task related concepts, the classes in green correspond to employee and responsibility related concepts, and the classes in orange correspond to rights and capability concepts and the classes in grey to governance rules concepts.

Concretely, the elaboration of the Responsibility metamodel has been performed following a design research method proposed by [2], named Action Design Research, which considers that the practitioners and end-users possess a rich knowledge regarding the research domain and that it is necessary to have them involved all along the artefact building activity. This method has consequently for objective to strengthen the connections between these practitioners and the researchers by combining the building, intervention and evaluation activities. Accordingly, it advocates for a continual evaluation of the problem and the built artefact (ReMMo) in order to ceaselessly adjust its elaboration with real usage settings. Therefore, a first version of ReMMo has been elaborated by analysing the literature from the fields of IT, requirement engineering, managerial and social sciences. Then, alpha versions have been iteratively generated in a limited organisational context. This evaluation by the practitioner was performed at the European Court of Auditors [3]. Afterwards, in a second step, the more mature artefact was evaluated in a wider organisational setting and beta versions were shaped with the end-users. This second iteration of ReMMo has been performed at the Hospital Center in Luxembourg [1].

1) Task and business object

As explained in i* [4], actors depend on each other to achieve a goal or to perform a task. In order to be compliant with these dependencies, while keeping the task as the unique concept concerned by the responsibility, we consider that both types of i* dependencies are Task types. To model this, we consider two types of attributes for the Task: the Goal and the Procedure and we express that one Goal always exists to define a Task although one Procedure may or may not exist [21, 26]. The business object is an object representing some concepts relevant to the organisation which are used by the Task [25]. Accordingly, we define the Task and the Business Object as:

DEFINITION 1: A task is a complete and identifiable piece of work necessary to achieve a goal and that may or may not be defined through a procedure. The task may be either a business task if it aims at achieving a business goal or a structural which if it aims at achieving a structural goal.

DEFINITION 2: A business object is a passive element (information, document or physical object) which has relevance from a business perspective and which may be used by one or many task(s).

2) Responsibility and Accountability, Actor, Sanction and Condition

Globally, most of the authors acknowledge that defining the responsibility aims at conferring one or more obligation(s) to an actor (the responsibility owner) [18-20]. As a consequence, that obligation provokes a moral or formal duty, in the mind of this responsibility owner, to justify the performance of the obligation to someone else. Beside the literature related to the responsibility, the review of the literature related to the accountability [27] highlights that the responsibility concerns a unique business task and aggregates a set of accountabilities which relates to this business task, to the task(s) needed by this business task, and to the structural task(s) concerned by this business task [21]. Accountability is broadly defined as the obligation to give account to someone else under the threat of sanction(s) [22] and is part of the responsibility [23-24]. Accordingly, we propose the following definitions:

DEFINITION 3: A responsibility is a charge assigned to a unique actor to signify its accountabilities concerning a unique business task.
DEFINITION 4: An accountability is an element which is part of a unique responsibility and which represents an obligation of an actor to achieve the goal, or to perform the procedure of a task, and the justification to someone else that it is done, under threat of sanction.

The responsibility is defined for a unique actor to which it is assigned. The concept of actor has already been largely defined in the literature [5] and it will not be reviewed in detail in this work. This concept of actor has been defined in i* as an active entity which carries out actions to achieve goals by exercising its know-how. This actor may be either an employee or a business role.

DEFINITION 5: An actor is an active entity which is assigned a set of responsibilities and that may check accountabilities.
DEFINITION 6: A business role is a type of actor which represents a set of employees who share common characteristics.
DEFINITION 7: An employee is a type of actor which represents a human entity which may or may not play one or more business roles.

Additionally, we have introduced the concepts of sanction and condition which we define as follows:

DEFINITION 8: A sanction is an element associated with an accountability and which corresponds to the consequence resulting from the justification of the realisation (or not) of this accountability.
DEFINITION 9: A condition defines a context which must be verified for the accountability to exist.

3) Capability and Right

To realise his accountability, an actor must possess a set of capabilities and rights to use business objects. These capabilities are intrinsic to the actor and correspond to the knowledge, the know-how, or the attitude he possesses. The concept of right is common but is not systematically embedded in all IT frameworks. It encompasses facilities required by an actor to discharge his accountability(ies). These rights are described in terms of accesses to a business object.
Capability and rights are components that have already been defined in the field of IT [6]. They have been introduced in ReMMo as well. These concepts are defined as follows:

DEFINITION 10: A capability represents the qualities, the skills or the resources intrinsic to the actor and which are required to discharge one or several accountability(ies).
DEFINITION 11: A right to use represents an authorisation to perform an operation on a business object which is required to discharge one or several accountability(ies).

4) Governance Rules and Source

In practice, we have observed that governance rules originated from governance sources. These governance sources provide high level rules that impact the elaboration of responsibilities by expressing conditions over the accountabilities. Both the governance rule and the source are defined as following:

DEFINITION 12: A governance rule is a high level prescript originating from dedicated sources and which constraints the definition of the accountabilities.
DEFINITION 13: A source is a formal piece of information which creates responsibilities and which contains, amongst other, required or desired governance rules.

B. ReMMo language

In previous section we have defined an expressive Responsibility metamodel aiming at supporting the modelling and the formalization of the responsibilities of employees. This Responsibility has been represented by means of an UML metamodel which is far from being user friendly and thereby, is difficult to read and to exploit in practice. For this reason, we have decided to integrate this responsibility metamodel with ArchiMate, an EA language, in order to benefit from the modelling language it provides. This integration benefits to the expressivity of ReMMo and consists in a responsibility extension of ArchiMate.

1) ArchiMate extension method

According to [7], the integration of two metamodels requires resolving three types of heterogeneities: syntactic, semantic and structural. For our integration, only the semantic and the structural heterogeneities have been addressed. Indeed, the syntactic heterogeneity aims at analyzing the difference between the serializations of metamodel and, as explained by [8], addresses technical heterogeneity like hardware

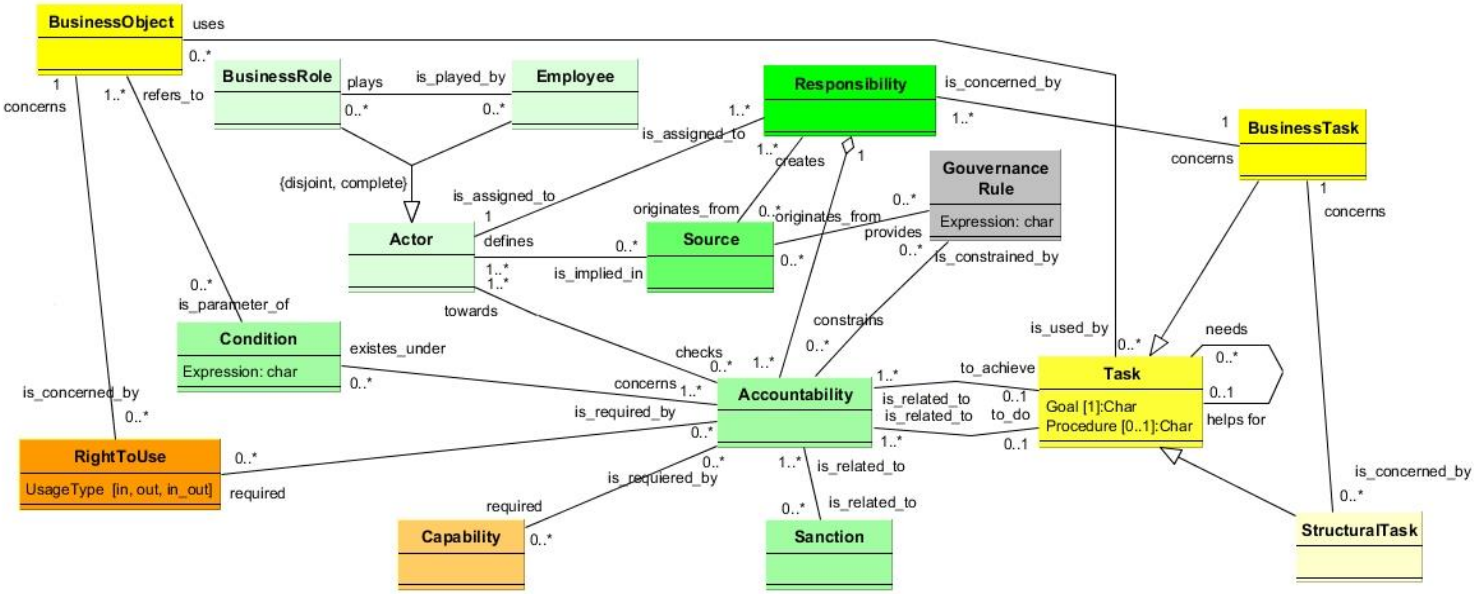

Figure 1. Responsibility metamodel UML Diagram

platforms and operating systems, or access methods, or it addresses the interface heterogeneity like the one which exists if different components are accessible through different access languages. The structural heterogeneity exists when the same metamodel concepts are modelled differently by each metamodel primitives. This structural heterogeneity has been addressed together with the analysis of the conceptual mapping and the definition of the integration rules. Finally, the semantic heterogeneity represents differences in the meaning of the considered metamodel' elements and must be addressed through elements mapping and integration rules. Regarding the mappings, three situations are possible: no mapping, a mapping of a type 1:1, and a mapping of a type n:m (n concepts from one metamodel are mapped with m concepts from the other). Practically, no case of n:m mapping has been encountered during the mapping between ReMMo and ArchiMate.

After defining the mapping, the concepts have been integrated in a single metamodel using both ArchiMate' extensions mechanisms: the addition of attribute and the specialization [9]. Concretely, if no mapping was detected, the concept from ReMMo was added in the ArchiMate using the first extension mechanism which consists in adding attribute to an existing concept. If a 1:1 mapping exists without conflict between two concepts, both concepts are merged in a unique one, this concept is added into the integrated metamodel, and this concept keeps the name of the ArchiMate concept. If a mapping of type 1:1 with conflict exists between two concepts, this means that one concept from one metamodel is richer or poorer than a concept from the other metamodel and in this case, both concepts are added in the integrated metamodel using the second extension mechanism of ArchiMate which is the stereotype (specialization).

2) ArchiMate Responsibility extensions

Concretely, in order to perform the mapping, it was necessary to remodel the concepts and the associations between concepts from ArchiMate in UML. ReMMo has been reworked as well in order to make the associations classes explicit and thereby model the mappings with the classes and relation classes from ArchiMate. The mapping realized between classes have been summarized in Table I. The mapping between relation classes is available in [1].

Table 1. Mapping of ReMMo with ArchiMate

Responsibility element	ArchiMate element	Mapping	Integration rule	Integrated element
Business Object	Business Object	1:1	Merge	Business Object
Task	Business Process	1:1	Specialisation	<<Task>>
R_Business Role[1]	Business Role	1:1	Specialisation	<<R_BusinessRole>>
Responsibility	Business Role	1:1	Specialisation	<<Responsibility>>
Employee	Business Actor	1:1	Specialisation	<<Employee>>
Accountability	Business Function	1:1	Specialisation	<<Accountability>>
Right To Use	Access association	1:1	Specialisation	<<RightToUse>>
Sanction	-	-	Addition of attribute	<<Accountability>>, Sanction: Sanction description
Condition	-	-	Addition of attribute	<<Accountability>>, Condition: Condition description
Capability	-	-	Addition of attribute	<<Accountability>>, Capability: Capability description
Source	Driver	1:1	Specialisation	<<Source>
Governance Rule	Requirement	1:1	Specialisation	<<Governance Rule>>

Two of them are illustrated in the following. As a first example of mapping, we have observed that the definition of the business role form ArchiMate which is "the responsibility for performing specific behavior, to which an actor can be assigned" [9] and the definition of the responsibility from ReMMo are semantically close but that the definition of the responsibility in ReMMo is more precise than the one from the business role in ArchiMate. Therefore, we have consider a 1:1 mapping with conflict between both concepts that have been added in the integrated metamodel and associated using a specialization link such that the Responsibility from ReMMo is a stereotype of the business role from ArchiMate written «Responsibility». A second example concerns the analysis of the definition of the concepts of business object from the ReMMo and from ArchiMate. Both definitions were semantically equivalent and both concepts have been merged in a unique one name Business Object.

[1] **R**_Business Role correspond to the Business Role from **R**eMMo

III. RBAC AND REMMO

The management of the access rights, based on RBAC and using the enterprise architecture approach, presents many potential advantages such as the possibility to align the access rights to be provided to the users, at the application layer, with the rights they really require at the business layer, to perform business processes. However, in practice, we notice that the concepts from the business layer of ArchiMate are roughly and imperfectly aligned with the concepts from the RBAC model, exploited at the application layer. This is mainly due to the lack of appropriate concepts, at the business layer, to precisely define and motivate the assignments of permissions to users. Given this weakness, ReMMo and the extension of ArchiMate with ReMMo presented in previous section could contribute to engineer and optimize the assignment of permissions to employees according to their responsibilities. Therefore, in Section 3.1, we remind how RBAC currently exists in, and may be modelled by, ArchiMate. To that end, we present the previous work realized by [10] related to the definition of a RBAC reference model at the application layer. Then, in Section 3.2, we analyze how the definition of the employee's responsibilities at the business layer could enhance the instantiation of RBAC at the application layer. Therefore, we align RBAC and the Responsibility metamodel, and we analyze which concepts from the Responsibility metamodel allow generating concepts from the RBAC model. Subsequently, based on this alignment, we propose an Access rights management reference model in Section 4. The latter includes, amongst others, five processes which contribute to populate the RBAC reference model proposed by Band.

A. RBAC reference model

Enterprise architecture practitioners acknowledge the advantage of modelling the business layer of the enterprise architecture according to the RBAC model to provide the business actors with accesses to the business objects [11]. At the application layer, RBAC has also been embedded in many operating systems and applications as well, [12]. This usage of the RBAC model at the application layer has also been corroborated by [10] who has proposed a RBAC reference model modelled by means of the existing core ArchiMate concepts. These concepts, which are exploited for this representation of RBAC at the application layer, are the data object which represents a passive element suitable for automated processing, and the application function which accesses the data object and represents a behavior element that groups automated behavior which can be performed by an application component [9]. To represent RBAC at the application layer, Band has created model to represent the management of the

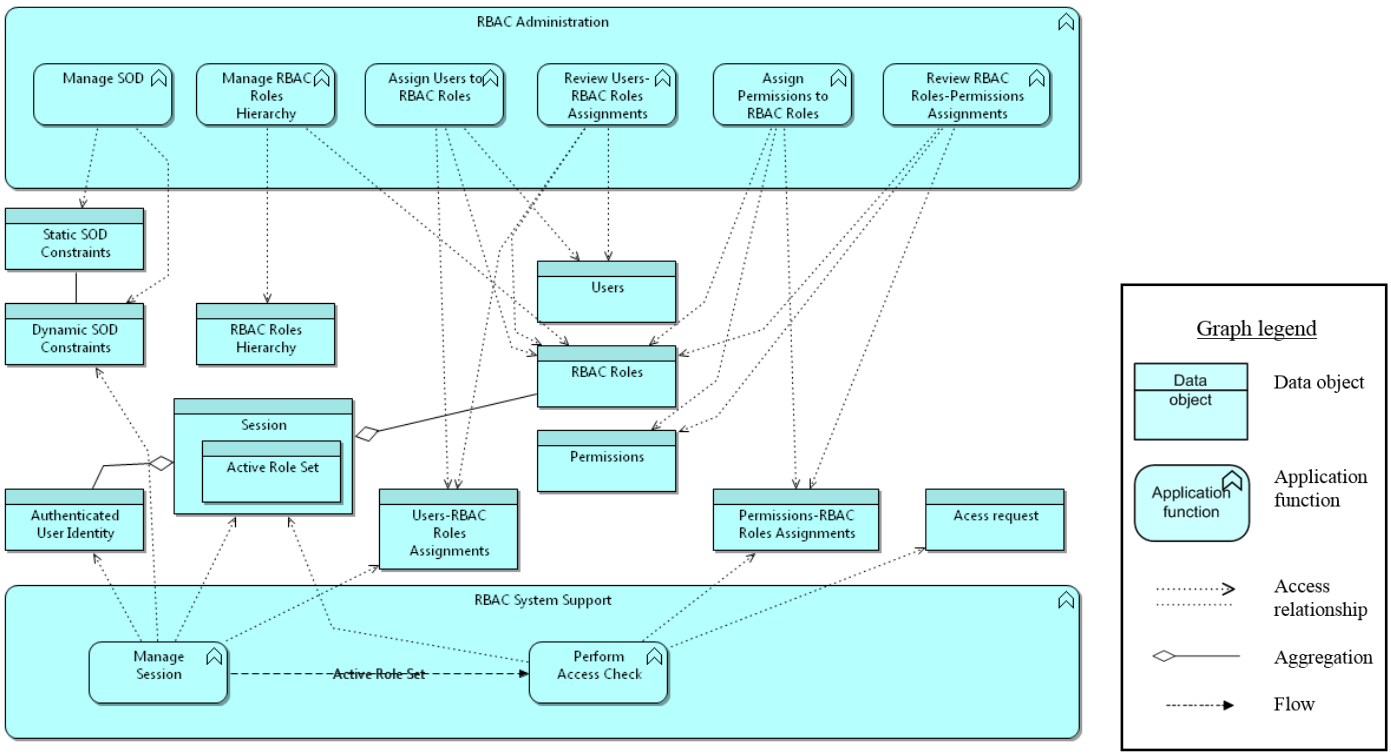

Figure 2. RBAC reference model in ArchiMate

access rights using ArchiMate and composed with a set of data objects and application functions which represent at the application layer, the concepts or associations between concepts from the business layer (Figure 2).

Practically, the administration of the access rights is performed by instantiating the data objects using the RBAC Administration main application function (Figure 2) and by performing access checks, according to the information represented by the data objects, using the *RBAC System Support* main function. Concerning the RBAC administration, this main function requests, on the one hand, to perform the assignment of Users to RBAC Roles, therefore the Assign Users to RBAC Roles application function reads the Users and the RBAC Roles data objects and writes the Users-RBAC Roles Assignments data object, and on the other hand, to execute the assignment of permissions to RBAC roles, therefore the Assign Permissions to RBAC Roles function reads the Permissions and the RBAC Roles data objects and writes the Permissions-RBAC Roles Assignments data object. For his part, the RBAC Support System allows checking that an access may be granted to the users by reading the Active Role Set and the Permissions to RBAC Roles Assignments, and by comparing this with the access requested.

B. ReMMo alignment with RBAC

This section aims at aligning ReMMo and RBAC with the objective not to elaborate an integrated metamodel but to figure out to what extend the elaboration of the responsibilities can be used to generate an RBAC model instance. The alignment is based on RBAC modelled in UML from [13-16]. In figure 3, the concepts from RBAC are represented in dark orange and the relations between concepts in light orange. The concepts from the Responsibility metamodel are in dark yellow and the relations between concepts in light yellow. To perform the alignment between concepts from RBAC and from the Responsibility metamodel, we exploit the "*trace to"* association. As defined by [17], the trace to association specifies the trace relationship between model elements or sets of model elements that represent a relationship between concepts in different models. Traces are mainly used for tracking requirements and changes across models. Since model changes can occur in both directions, the directionality of the dependency can often be ignored. The mapping specifies the relationship between the two, but it is rarely computable and is usually informal. The following trace to relationships between concepts, and relations between concepts, are identified (figure 3.):

- The employee from the Responsibility metamodel is defined as a human entity that may or may not play one or more business roles. Depending on the business role played, the employee may require permissions on the information system. In RBAC, the user mainly represents a human. There exists a trace to association between the User class from RBAC and the Employee class from ReMMo.

The business role from the Responsibility metamodel may represent a set of employees who share common characteristics and are assigned to responsibilities. This business role may or may not require permissions on the information system. In RBAC, the role means a job function with some associated semantics regarding the responsibilities conferred to the users assigned to it. There exists a trace to association between the concept of RBAC Role from RBAC, and the BusinessRole class from the Responsibility metamodel.

- In ReMMo, the employee is associated to the business role through the Play association. In the RBAC model, the user is associated to the RBAC role. There exists a trace to association between the Play association and Users-RBAC Roles Assign association classes. However, a RBAC role, generated from a business role is assigned to a user which is generated from an employee only if this business role is played by this employee. Therefore we introduce the constraint A.I which is: *A RBAC Role generated from a BusinessRole is assigned to a User generated from an Employee if this BusinessRole is played by this Employee.*

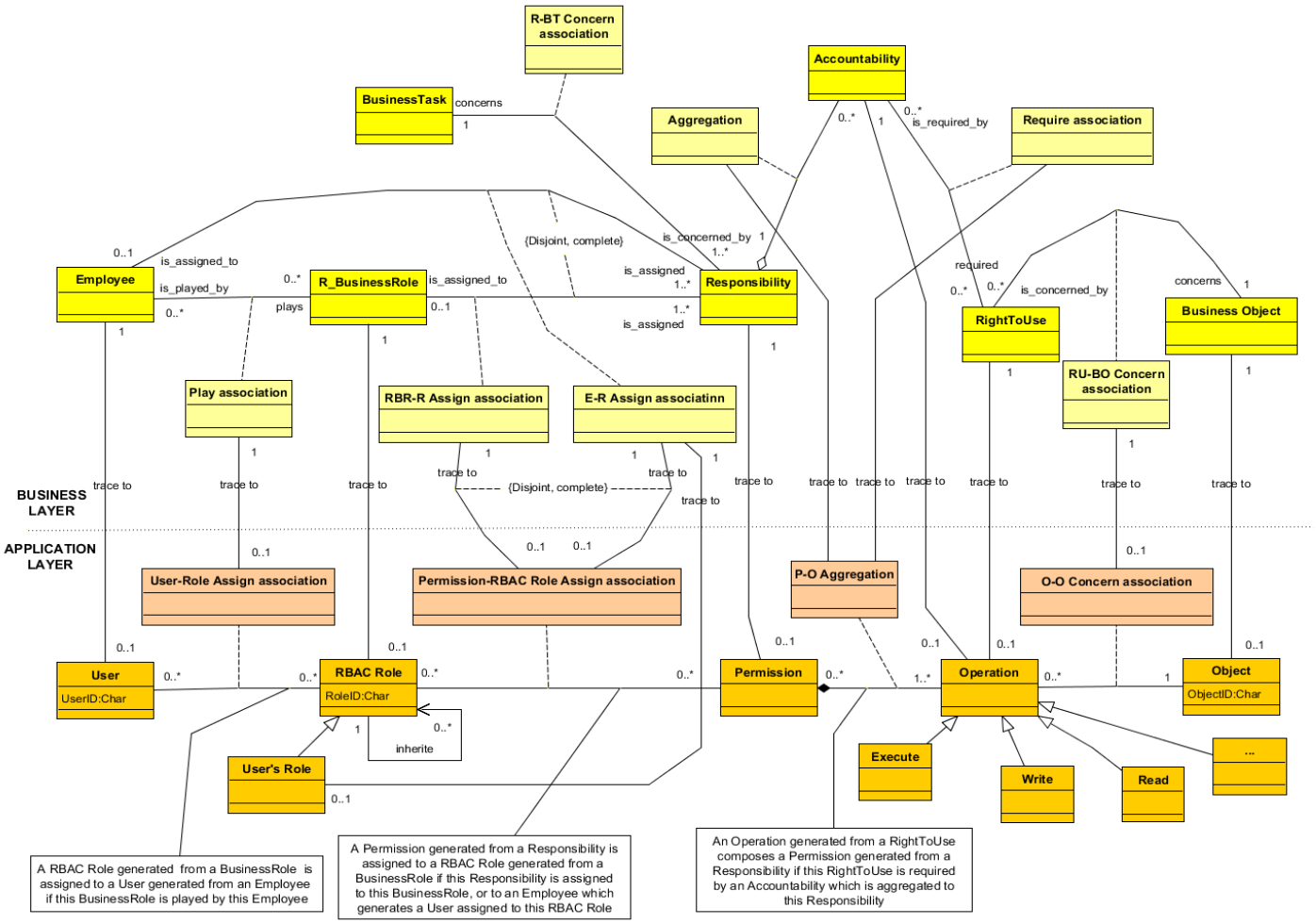

Figure 3. Alignment between RBAC and ReMMo

- In ReMMo, the employee is associated to the business role through the Play association. In the RBAC model, the user is associated to the RBAC role. There exists a trace to association between the Play association and Users-RBAC Roles Assign association classes. However, a RBAC role, generated from a business role is assigned to a user which is generated from an employee only if this business role is played by this employee. Therefore we introduce the constraint A.I which is: *A RBAC Role generated from a BusinessRole is assigned to a User generated from an Employee if this BusinessRole is played by this Employee.*

- In the Responsibility metamodel, the responsibility concerns a unique business task and aggregates a set of accountabilities which relates to this business task (Cf. Section 2.1.2). These accountabilities require rights to use the business objects which are, themselves, used by the business task. In RBAC, according to [14] a permission determines which operations a user assigned to a role can perform on information resources. Hence a permission encompasses a set of operations on business objects. Both the responsibility and the permission from RBAC have in common the gathering of a list of operations related to business objects. As a consequence, we observe a trace to association between the Permission class and the Responsibility class, as represented in figure 3. Acknowledging this mapping, we observe that while there exists no justification nor guideline, in RBAC, for the gathering of a set of operations within a permission, the mapping of RBAC with ReMMo permits to justify that these operations are gathered according to the unique business task concerned by the responsibility.

- In the Responsibility metamodel, an employee may be directly assigned to a responsibility although in RBAC, a user may not directly be assigned to a permission. Practically, to realize this association with RBAC, we need to define a special RBAC role such as only the user which corresponds to the employee directly assigned to the responsibility is assigned to this role. This special RBAC role is named User's Role, is represented by the User's role class, in figure 3, and corresponds to a specialization of the RBAC Role class. This User's Role is generated to represent the E-R Assign association class. Therefore, we associate both concepts with a trace to association.

- In the Responsibility metamodel, the responsibility is associated to the business role or to the employee through, respectively, the RBR-R Assign association or the E-R Assign association. In the RBAC model, the permission is associated to the RBAC role through the Permission-RBAC Role Assign association. We observe that this RBAC role is generated either by the RBR-R Assign association or by the E-R Assign association but not by both associations at the same time. Therefore, we create two trace to associations: (1) between the Permission-RBAC Role Assign association class from RBAC and the RBR-R Assign association class from the Responsibility metamodel and (2) between the Permission-RBAC Role Assign association class from RBAC and the E-R Assign association class from the Responsibility metamodel, and we express the constraint that these

relations are Disjoint and Complete. Additionally, we observe that a permission, generated from a responsibility, is assigned to a RBAC role, generated from a business role, if this responsibility is assigned to this business role or to an employee which generates a user assigned to this RBAC role. This is expressed by the constraint: *A Permission generated from a Responsibility is assigned to a RBAC Role generated from a BusinessRole if this Responsibility is assigned to this BusinessRole, or to an Employee which generates a User assigned to this RBAC Role*.

- In the Responsibility metamodel, the right to use corresponds to an authorization to perform an operation on a business object. In RBAC, a permission is defined as an approval of a mode of access to a resource. Hence, we consider that there exists a trace to association between the RightToUse class from ReMMo and the Operation class from RBAC.

- In RBAC, an object corresponds to an information object. In ReMMo, the business object is defined as a passive element which may be a document, an information or a physical object. We consider that there exists a trace to association between the Business Object class from the Responsibility metamodel and the Object class from RBAC, but not for a physical object.

- Concerning the associations, we observe that the BO-R Concern association between the business object and the right to use from ReMMo generates the O-O Concern association between the operation and the object from RBAC. We represent this by a trace to association between the BO-R Concern association class from the Responsibility metamodel and the O-O Concern association class from RBAC.

- Finally, in the Responsibility metamodel, the responsibility aggregates accountabilities which require rights to use, although in RBAC, the permission aggregates operations. Practically, in the Responsibility metamodel, this is represented by the Aggregation and the Require associations, and in RBAC this is represented by the P-O Aggregation. We observe that the latter is generated by the Aggregation between the responsibility and the accountability, and by the Require association between the accountability and the right to use, from ReMMo. Therefore, we create two trace to associations. The first one is between the P-O Aggregation class and the Aggregation class and the second one is between the P-O Aggregation class and the Require association class. Additionally, we observe that an operation generated from a right to use composes a permission generated from a responsibility if this right to use is required by an accountability which is aggregated to this responsibility. This is expressed by the constraint: *An Operation generated from a RightToUse composes a Permission generated from a Responsibility if this RightToUse is required by an Accountability which is aggregated to this Responsibility*

IV. ACCESS RIGHTS MANAGEMENT REFERENCE MODEL

This section proposes an *Access rights management reference model*, at the business layer, and illustrates: (1) how this access rights management may be decomposed into five processes, and (2) how these processes write a set of dedicated business objects which are afterwards realized by data objects used to represent and relate the users, the RBAC roles and the permissions at the application layer. The population of these business objects is performed by the *RBAC administrator* which collects the information related to the *employees*, the *business roles*, and the *responsibilities* assigned to both, from the analysis of the business layer. Practically, this business layer is described and analyzed through business processes documentation, job descriptions, or interviews of employees and managers.

Considering the integration of ReMMo with the business layer of ArchiMate performed in Section 2.2.2 and given the alignment of the Responsibility metamodel with RBAC performed in Section 3.2, the *Access rights management reference model* aims to populate and to extend elements from the *RBAC reference model* proposed in Section 3.1 according to the responsibilities of the employees defined at the business layer of ArchiMate. This *Access rights management reference model* is presented in Figure 4. The lower layer of it represents a fragment of the *RBAC reference model*, at the application layer, and the upper layer represents, at the business layer, the *Access rights management reference model* itself (this layer is named: Access Rights Management). The concepts from the *RBAC reference model* which are represented and which we want to instantiate are the *users*, the *RBAC roles*, the *permissions*, the *users-RBAC roles assignments* and the *permissions-RBAC roles assignments*. The *Access Rights Management layer* represents the access rights management processes which collect the information from the responsibilities of the employees, modelled with ArchiMate extended with ReMMo. This access rights management layer is composed of the *RBAC administrator role*, which is assigned to five business processes:

- The first process, necessary to manage the access rights, populates the list of users. The user is a person from the subset of employees who requires to use the information system. We have analyzed that the users are generated from the employees. As a result, this process aims at collecting, at the business layer of the enterprise, the list of employees who need to access the information system in order to perform the business process they are assigned to. Hence, in practice, this list of employees is collected from the responsibilities modelled with ArchiMate extended with the Responsibility metamodel. The result of the deployment of this process is a business object named *List of Users*. This is represented in Figure 4 by the *Populate the list of Users* process writes the *List of Users* business object. Afterwards, to be handled by the *RBAC reference model* at the application layer, this *List of User* business object is realized by the *Users* data object. This is represented by the *Users* data object *realize* the *List of Users* business object. The *realize* association from ArchiMate is represented by dashed line arrow in Figure 4.

- The second process concerns the population of the list of RBAC roles. The RBAC role is a role from the subset of business roles which requires to use the information system. Given that the RBAC roles are generated from the business roles, this process aims at collecting, at the business layer of the enterprise, the list of business roles which need to access the information system. Hence, similarly to what we have done for the employees, this list of business roles is collected from the responsibilities modelled with ArchiMate extended with the Responsibility metamodel. The result of the deployment of this process is a business object named *List of RBAC Roles*. This is represented in Figure 4 by the *Populate the list of RBAC Roles* process that writes the *List of RBAC Roles* business object. Afterwards, to be handled by the *RBAC reference model* at the application layer, this *List of RBAC Roles* business object is *realized* by the *Roles* data object. This is represented by the *Roles* data object *realize* the *List of RBAC Roles* business object.

The third process populates the list of permissions. Given the alignment of RBAC with ReMMo, we have analyzed that the permissions are generated from the responsibilities assigned to the employees or to the business roles regarding a specific business task. As a result, to populate the list of permissions, the RBAC administrator must collect, through the responsibilities modelled by the ArchiMate extended with the Responsibility metamodel, and the rights to use required to realize the responsibilities related to a specific business task. The result of the deployment of this process is a business object named *List of Permissions*. Practically, this is represented by the *Populate the list of Permissions* process writes the *List of Permissions* business object. Equally, to be handled by the RBAC reference model, this *List of Permissions* business object is realized by the *List of Permissions* data object. This is represented by the *Permissions* business object *realize* the *List of Permissions* data object.

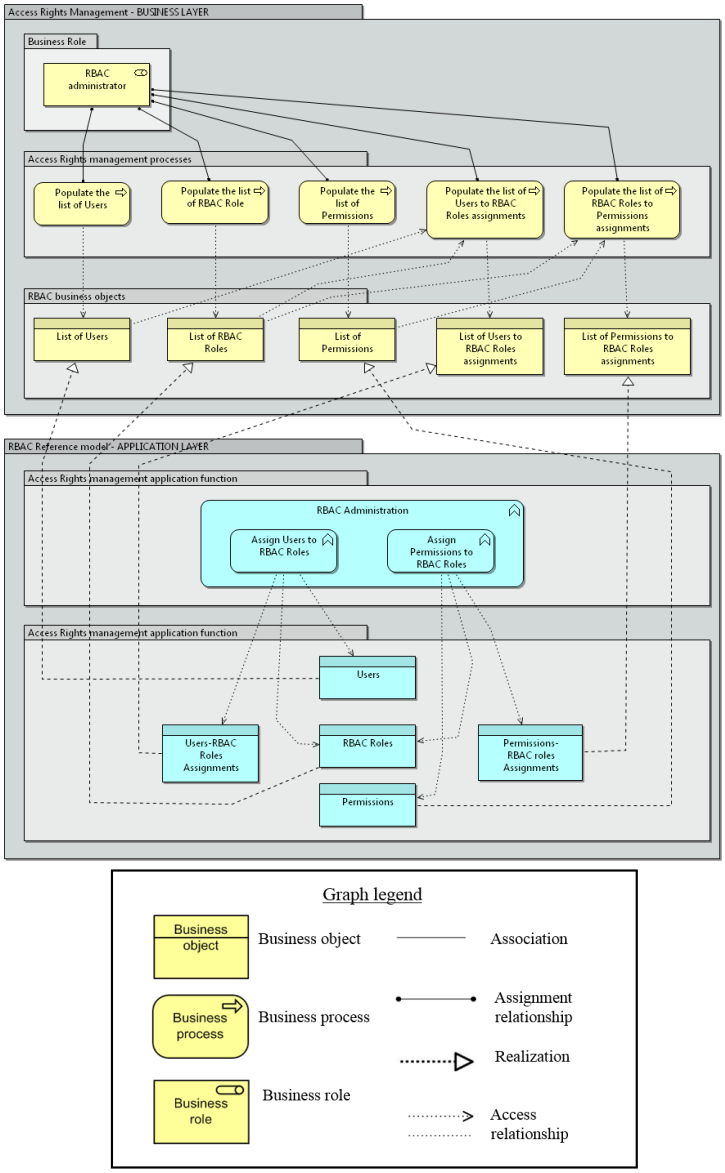

Figure 4. Access rights management reference model

- The fourth process populates the list of users assigned to RBAC roles. This process analyses the responsibilities modelled at the business layer of the enterprise in order to formalize the List of Users to RBAC Roles assignments which is a business object which represents which employee (who requires to use information at the application layer) plays which business role. At the business layer, this is modelled using the business object *List of Users to RBAC Roles assignments* which is related to the *Populate the list of Users to RBAC Roles assignments* process by means of a write relation. Additionally, to be handled by the *RBAC reference model*, at the application layer, *this List of users to RBAC roles assignments* business objects is realized by the *Users-RBAC Roles Assignments* data objects which is represented by the *Users-RBAC Roles Assignments* business object *realize* the *Populate the list of Users to RBAC Roles assignments* data object.

- The fifth and last process populates the list of permissions assigned to RBAC role. In the same way, this process analyses the responsibilities modelled in the business layer to formalize the List of Permissions to RBAC Roles assignments which is a business object which represents which permissions are required by the business role in order to use information existing at the application layer. Considering the alignment of RBAC with the Responsibility metamodel, this list of permissions to RBAC roles assignments is populated from the association of responsibilities to business roles or to employees. At the business layer, this is modelled using the business object *List of Permissions to RBAC Roles assignments*. This business oject is related to the *Populate the list of Permissions to RBAC Roles assignments* process by means of a write relation. Additionally, to be handled by the *RBAC reference model*, this *List of permissions to RBAC roles assignments* business objects is realized by the *Permissions-RBAC Roles Assignments* data objects. This is represented by the *Permissions-RBAC Roles Assignments* business object *realize* the *Populate the list of Permissions to RBAC Roles assignments* data object.

The next section of this paper illustrates the method for the access rights management based on the Access rights management reference model using the second part of the case study in the Hospital Center in Luxembourg.

V. CASE STUDY

The context of the case study is the access rights management at the *Hospital Center in Luxembourg*. The two following objectives are targeted: evaluate that the integrated ArchiMate with ReMMo, at the business layer, enhances the definition of the access rights required by the business role, and evaluate that the definition of the responsibilities at the business layer may be used to generate the RBAC roles and permissions, at the application layer. In practice, this is illustrated with the *Receptionist role* from the hospital.

This case study was realized during the months of February and March 2012. During this period, four meetings of two hours were organized with the *Reception department manager*, and the *Manager of the competences*. During these meetings, we have analyzed the *Receptionist role*, defined the responsibilities, and analyzed the rights to use required for each of the accountabilities aggregated by these responsibilities. Therefore, both employees have provided the information necessary to understand the responsibilities of the staff working in the reception department and the *Applications Support Engineer IT Services* has provided the list of existing RBAC roles and permissions.

The case study is structured as follows. In Section 5.1, we analyze the existing rights management activity in the hospital and we extract the list of permissions actually assigned to business roles. In Section 5.2, we deploy the method based on the Access rights management reference model and we define new permissions to be assigned to the same business roles. Finally, in Section 5.3, we compare the permissions provided to those really required to draw conclusions.

A. Existing access rights management in the hospital

In practice, in the hospital, when a new employee is hired for the reception, the department manager assigns him, at the business layer, to one of the eight business roles (**BR**) and, at the application layer, according to the *RBAC administrator*, to a set of RBAC roles (**RR**).
The business roles are:

- BR1: Receptionist at *Clinic of Eich* and Municipal hospital
- BR2: Receptionist at the pediatric clinic and maternity
- BR3: Phone reception
- BR4: Infodesk
- BR5: Human resources management
- BR6: Department management
- BR7: Room operator
- BR8: Outsourced guardian
- REFR=∑ of permissions assigned to RR1, RR2 and RR3

The application roles are assignment based on, and supported by, an existing list of correspondences between both types of roles. The objective of these correspondences is to facilitate the assignment of RBAC roles to the employee depending on the business roles they are assigned to. This correspondence between the business role and the RBAC role is equivalent to the trace to association illustrated in Figure 4. This correspondence is explained in Table II.

Table II. Correspondence existing Business roles / RBAC roles.

Business Role	**RBAC roles**
BR1	REFR, RR6
BR2	REFR, RR4, RR5, RR6
BR3	REFR, RR6, RR7
BR4	REFR, RR6
BR5	REFR, RR4, RR5, RR6, RR8, RR11
BR6	REFR, RR4, RR5, RR6, RR7, RR8, RR9, RR10, RR11
BR7	RR10
BR8	RR6, RR9

Each RBAC role is associated to a set of permissions defined by a set of operations on a data object. Eg. **RR2** is assigned to the permission to **Create, add, modify, display, and delete** *the Bed status file* or **RR7** to the permission to **Display** the *Planning of doctors on duty file*. The complete list is available in [1].

B. Analysis of the access rights really required by the business roles

In this section, we analyse the permissions which should be provided to the business roles according to the responsibilities they are assigned to, for the reception of the hospital. Therefore, we exploit the method defined in Section 4 which aims at performing the processes which compose the Access right management reference model to instantiate the RBAC reference model, at the application layer. In the hospital, as reviewed in previous sections, the permissions are exclusively assigned to the business roles and not to the employees. The two following processes are not considered: *Populate the list of Users* and *Populate the list of User to RBAC Roles assignments*.

1) Population of the list of RBAC roles

Given that the RBAC roles are generated from the business roles (Section 3.2), the *Populate the list of RBAC Roles* process firstly needs to analyze the business layer of the hospital to model the business roles which need to access the information system and secondly, generate the RBAC roles from these business roles. In practice, the Human Resources (HR) department of the hospital is involved in the definition of the *Job descriptions*. These job descriptions aim at describing the tasks to be performed by each business role, as well as the necessary required knowledge associated to it. However, the job descriptions do not specify the required access rights on professional software. Using this document, for the reception department, 8 business roles have been detected and correspond to those listed in Section 5.1. In this case study, we analyze the permissions really required by the business roles to compare them with the permissions they really received. To compare the same things, we conserve the same business roles as the one used in this Section 5.1. Additionally, according to the alignment between the Responsibility metamodel and the RBAC model explained in Figure 3, zero to one RBAC role trace to one BusinessRole. Therefore, the result of this step is 8 RBAC roles whish correspond exactly to BusinessRole. The case of an Employee corresponding to a User's role has not been encountered.

2) Population of the list permissions

To populate the list of permissions, according to the alignment of ReMMo with RBAC, the first step consists in modelling the responsibilities which are assigned to the business roles, the accountabilities that are aggregated by these responsibilities and the rights to use required by these accountabilities. These responsibilities and accountabilities do not formally exist in the hospital but may be engineered from the Job description related to the receptionist's role analysis. Regarding the rights to use, they do not exist in the Job description but may be discovered by interviewing the reception manager.
To model the responsibilities, the RBAC administrator must analyze the business layer of the enterprise, and in the case of the hospital, the information provided by the *Job description*. In our case, sixteen responsibilities have been extracted for the reception. The responsibilities 6 and 14 (partially) are presented in this case study in Figure 5. The complete set of responsibility is available in [1].
As illustrated on this Figure, the *Department Manager* is assigned to R*esponsibility 14* which aggregates two accountabilities, firstly, the accountability *to do the management of the reception* which requires the RightToUse of a type **Write** the business object *Room agenda* and the RightToUse of a type **Read/write** the business objects *Equipment ordering, the Reception planning, the Infrastructure report and the Statistics*, and secondly, the accountability *to achieve the creation and modification of the patient's invoices* which requires the right to use of a type **Read** the business object *Patient's invoices*.

3) Population of the list permissions assigned to RBAC roles

To populate the list permissions assigned to RBAC roles, according to the alignment of the Responsibility metamodel and the RBAC model, the process firstly needs to model the responsibilities assigned to business roles. Therefore, the RBAC administrator again needs to analyze the business layer of the reception.

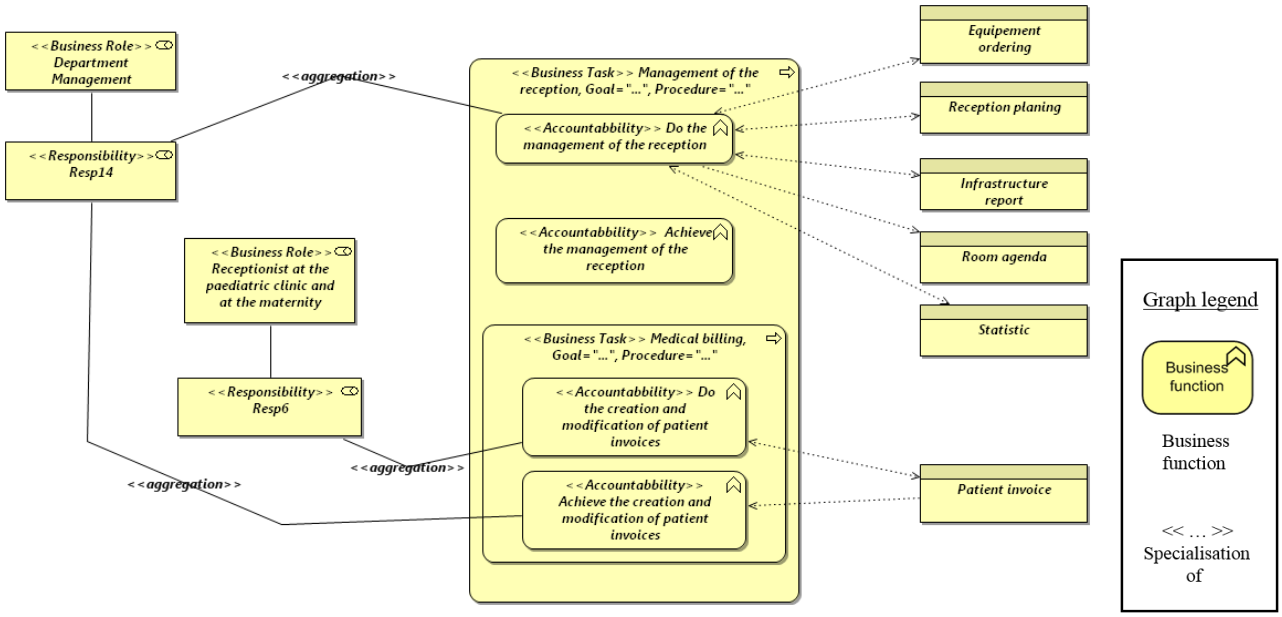

Figure 5. Example of responsibility modeled following the ArchiMate formalism

In our case, we have analysed the *Job description* and we have interviewed the *Reception department manager*. Based on the collected information, we have defined set of associations between the business roles and the responsibilities assigned to the latter. Eg.: the **BR6** (Department Manager) is composed of the responsibilities 14 (Figure 5), **BR1** (Receptionist at the Clinic of Eich and at the Municipal hospital) is composed of the responsibility 1, 2, 3 and 7 (Cf. [1]).

C. Case study analysis and discussion

By comparing the access rights actually assigned to the business roles (Section 5.2) and the ones engineered using the responsibilities (Section 5.3), we have observed the following not required existing permissions (Table III):

- **BR3** and **BR4** are granted too many permissions. Actually, the employees assigned to the *Phone reception role* and to the *Infodesk role* are authorised to **Create, add, modify, display, delete** the *Basic patient's data and entry, transfer, or leaving data* although they do not require these permissions. Additionally, they are assigned to the permissions **Create, add, modify and delete** the *Bed status files* although they only require the permission to display the bed status file.

- **BR1**, **BR2**, **BR5** are not assigned to the responsibility which aggregates the accountability *to do equipment ordering*, although they have the permission to **Create, add, modify, display and delete** *Data in the equipment ordering software*.

- **BR6** is actually granted all the permissions assigned to the other BR's. This was motivated by the fact that the *Reception management role* must supervise and monitor the other business roles. However, by analyzing the responsibilities assigned to this business role, we have observed that the responsibility 14 is composed of the accountability *to do the management of the reception* and the accountabilities *to achieve all the other business tasks* of this department. In practice, the accountabilities to achieve tasks only require to read information in order to monitor the business task and not to write the information such as it is actually defined.

- **BR8** is granted the right to **Delete** *Data in the reporting software* although none of the responsibilities assigned to this business roles aggregates accountabilities which require such a permission.

Although the case study has allowed demonstrating that the access rights are more accurately determined using ReMMo, it does not allow drawing conclusions concerning the operationalization of the method in a larger environment.

VI. RELATED WORKS

The review of the related works shows that two types of approaches coexist regarding the roles and rights engineering methods: the top-down and the bottom-up [36]. This related works section has been restrained to the analysis of top-down solutions which exploit the concepts existing at the application or business layers of the organisation. This means that the other solutions such as those based on roles mining have not been considered. These solutions aim at utilizing the existing permission assignments to formulate roles [37]. Starting from the existing permissions before RBAC is implemented, the bottom-up approach aggregates these into roles. This restriction is mainly justified by the fact that the top-down approaches traditionally do not recognise the existing permissions although the bottom-up does not consider business concepts from the organisation, which is in opposition to our research objective regarding the enhancement of the business/IT alignment. The existing top-down methods that we have analyzed are those that we estimate the more suitable to motivate the objectives of our research. The Role Finding approach [30] proposes (1) a method based on a process-oriented approach to define roles. In this method, a metamodel is built upon 3 layers: processes, roles and access rights and (2) a procedural model to express the steps for instantiating the concepts of the process layer. Crook et al. proposed the Analytical Role Modelling Framework in [31] and in [32] to model roles following the RBAC model together with the definition of the links with the organisational structure. In [31] exploits i* to model the relations between actors using the Strategic Rationale (SR) model and

to model access policies that take into account the organisational context. [32] makes the link between the RBAC model and the SR model to derive the roles from the actors and the permissions from the tasks. [33] describes a 5 steps methodology to define an access control service for an information system in the field of health. Thereafter, he defines the Dynamic Authorisation Framework for Multiple Authorisation Types which is composed of a hybrid access control model and of a logic-driven authorization engine. R/PAM (Role/Permission Assignment Model) is a model proposed by Epstein which permits to demonstrate that it is possible to decompose roles into permissions or to aggregate permissions into a role [29]. [34] explains that one method for determining functional requirements is the definition of uses cases. They propose a method to determine the needs for a role considering use cases and sequences of the use cases and define the authorisation rules based on all the use cases of the system. [35] proposes a role engineering method based on the scenario model and argue that the scenario can be considered as a set of steps on which particular access operations are associated

Globally, most of the methods tend to engineer the access rights with the application layer and considering specific access control models. These methods most often exploit the functional requirements analysis, e.g. [30, 31, 35], and are mostly dedicated to the definition or the instantiation of the RBAC model, e.g. [30, 31, 32, 35].

Table III. List of differences between existing and required rights

Business role	Exiting Permissions	Required Permissions	Not Required Existing Permissions
BR1	**Create, add, modify, display, delete** *the basic patient's data, the entry, transfer or leaving of patient's data, the bed status file, and data in the equipment ordering software*	**Create, add, modify, display, delete** *the basic patient's data, the entry, transfer or leaving of patient's data, the bed status file, and data in the equipment ordering software*	**Create, add, modify, display, delete** *data in the equipment ordering software*
BR2	**Create, add, modify, display, *delete*** *the basic patient's data, the entry, transfer or leaving of patient's data, the bed status file, the medical delivery data, and data in the equipment ordering software,* **Create, add, modify, display** *the patient's invoices record*	**Create, add, modify, display, delete** *the basic patient's data, the entry, transfer or leaving of patient's data, the bed status file, the medical delivery data,* **Create, add, modify** *the patient's invoices record*	**Create, add, modify, display, delete** *data in the equipment ordering software*
BR3	**Create, add, modify, display, delete** *the basic patient's data, the entry, transfer or leaving of patient's data, the bed status file, and data in the equipment ordering software,* **Display** *the planning of doctors on duty file*	**Create, add, modify, display, delete** *data in the equipment ordering software,* **Display** *the bed status file and the planning of doctors on duty file*	**Create, add, modify, display, delete** *the basic patient's data and the entry, transfer or leaving of patient's data,* **Create, add, modify, delete** *the bed status file*
BR4	**Create, add, modify, display, delete** *the basic patient's data, the entry, transfer or leaving of patient's data, the bed status file, data in the equipment ordering software*	**Create, add, modify, display, delete** *data in the equipment ordering software,* **Display** *the bed status file*	**Create, add, modify, display, delete** *the basic patient's data and the entry, transfer or leaving of patient's data,* **Create, add, modify, delete** *the bed status file*
BR5	**Create, add, modify, display, delete** *the basic patient's data, the entry, transfer or leaving of patient's data, the bed status file, the medical delivery data, data in the equipment ordering software, data in the Excel file: Timetable planning, and data in the statistical software,* **Create, add, modify, display** *the patient's invoices record*	**Create, add, modify, display, delete** *data in the Excel file: Timetable planning, and data in the statistical software*	**Create, add, modify, display, delete** *the basic patient's data, the entry, transfer or leaving of patient's data, the bed status file, the medical delivery data, data in the equipment ordering software,* **Create, add, modify, display** *the patient's invoices record*
BR6	**Create, add, modify, display, delete** *the basic patient's data, the entry, transfer or leaving of patient's data, the bed status file, the medical delivery data, data in the equipment ordering software, data in the Excel file: Timetable planning, data the room agenda in GroupWise multi-users software, and data in the statistical software,* **Create, add, modify, display** *the patient's invoices record, the planning of doctors on duty file, and the data in the reporting software*	**Display** *the basic patient's data, the entry, transfer or leaving of patient's data, the bed status file, the medical delivery data, the patient's invoices record and the planning of doctors on duty file,* **Create, add, modify, display, delete** *data in the Excel file: Timetable planning, data the room agenda in GroupWise multi-users software, and data in the statistical software, data in the equipment ordering software,* **Create, add, modify, display** *data in the reporting software*	**Create, add, modify, delete** *the basic patient's data, the entry, transfer or leaving of patient's data, the bed status file, the medical delivery data,* **Create, add, modify, display** *the patient's invoices record*
BR7	**Create, add, modify, display, delete** *data the room agenda in GroupWise multi-users software and data in the statistical software*	**Create, add, modify, display, delete** *data in the room agenda in GroupWise multi-users software and data in the statistical software*	
BR8	**Create, add, modify, display, delete** d*ata in the equipment ordering software and data in the reporting software,* **Display** *data from the room agenda in Group- Wise multi-users software*	**Create, add, modify, display, delete** *data in the equipment ordering software,* **Create, add, modify** *data in the reporting software,* **Display** *data from the room agenda in GroupWise multi-users software*	**Delete** *data in the reporting software*

VII. CONCLUSION AND FUTURE WORKS

In this paper, we have first presented an expressive Responsibility metamodel that we have mapped the ArchiMate to enhance is usability. This mapping allows one to model responsibility using the EA metamodel and thereby, enhances the associations between the business concepts of business actor, business role, business process, business function and business object have been semantically enriched. This integrated metamodel allows refining the responsibilities of employees and assigning access rights to them considering the accountabilities which compose the responsibilities. Afterwards we have aligned ReMMo and the RBAC. This alignment has resulted in the definition of a set of "trace to" associations between the concepts from both latter such as the *User trace to Employee*, the RBAC Role trace to Business Role, the Permission trace to Responsibility, the RightToUs trace to the Operation and the BusinessObject trace to the Object. Then we have proposed a method to populate, based on this alignment, the RBAC reference model existing at the application layer.

To illustrate and evaluate this alignment and the method, we have introduced the case study in the *Hospital Center in Luxembourg*. This case study aimed at improving the alignment of the business layer of the hospital with its application layer. Therefore, the human resources department defines Job descriptions which formalize the responsibilities to be achieved by the business roles. The results of this case study were that seven business roles over eight are granted too many permissions and, hence, that using the ReMMo to define the access rights allows enhancing the alignment between the enterprises layers.

Two future works are identified: first, the development of a tool to support the deployment of the *Access rights management reference model*, and second, the mapping of ReMMo with other access control models like, for instance, the ABAC model [28].

REFERENCES

[1] Feltus, C. Aligning access rights to governance needs with the responsibility metamodel (ReMMo) in the frame of enterprise architecture, 2014, University of Namur, Belgium

[2] Sein, M.K., Henfridsson, O., Purao, S., Rossi, M., and Lindgren, R. Action design research. MIS Q., 35(1):37-56, March 2011.

[3] Petit, M., Feltus, C.,Vernadat, F. Enterprise Architecture Enhanced with Responsibility to Manage Access Rights - Case Study in an EU Institution, PoEM 2012, Rostock, Germany.

[4] Eric S.K.Y. Towards modeling and reasoning support for early-phase requirements engineering. RE '97, Washington, DC, USA, 1997. IEEE.

[5] Amyot, D., Horkoff, J., Gross, D., and Mussbacher, G. A lightweight GRL profilefor i* modeling. ER 2009 Workshops on Advances in Conceptual Modeling – Challenging Perspectives, Berlin, Heidelberg, 2009.

[6] Vernadat, F. Enterprise modelling and integration. In ICEIMT, pages 25-33. 2002.

[7] Parent, C. and Spaccapietra, S. Database integration: The key to data interoperability. Advances in Object-Oriented Data Modeling, 2000.

[8] Zivkovic, S., Kühn, H., and Karagiannis, D. Facilitate modelling using method integration: An approach using mappings and integration rules. ECIS 2007, pages 2038-2049. University of St. Gallen.

[9] The Open Group. ArchiMate® 2.1 Specification. Van Haren Publishing, The Netherlands. 2012-2013.

[10] Band, I. Modeling rbac with sabsa, togaf and archimate. In The Open Group Conference, Austin, Texas. 2011.

[11] Gaaloul, K. and Proper, H.A.E. An access control model for organisational management in enterprise architecture. In Proceedings of the 9th International Conference on Semantics, Knowledge and Grids. 2013.

[12] Cenys, A. Normantas, A., and Radvilavicius, L. Designing role-based access control policies with uml. Journal of Engineering Science and Technology Review, 2(1), 2009.

[13] Shin, M.E., and Ahn, G.-J. Uml-based representation of role-based access control, WETICE '00, pages 195-200, Washington, DC, USA, 2000.

[14] Ray, I., Li, N., France, R., and Kim, D.K., Using uml to visualize role-based access control constraints. SACMAT '04, NY, USA, 2004. ACM.

[15] Kim, D.K., Ray, I., France, R., and Li, N. Modeling role-based access control using parameterized uml models. FASE, vol. 2984. Springer, 2004.

[16] Anderson, A. Xacml profile for role based access control (rbac). Technical Report Draft 1, OASIS, February 2004.

[17] Object Management Group (OMG). Uml 2.4.1 superstructure specification. 2011.

[18] Storer, T., Lock, R. Modelling responsibility. Project working paper 7, indeed project. 2008.

[19] Sommerville, I. Models for responsibility assignment. Responsibility and Dependable Systems 165-186, Springer. 2007.

[20] Strens, R., and Dobson, J. How responsibility modelling leads to security requirements. NSPW, 143-149, NY, USA. 1993, 143-149.

[21] Feltus, C., Petit, M. and Dubois, E. Strengthening employee's responsibility to enhance governance of IT: COBIT RACI chart case study. WISG '09. ACM, New York, NY, USA, 23-32.

[22] Blind, P.K. Accountability in public service delivery: A multidisciplinary review of the concept. Vienna, Austria. 2001.

[23] Bovens, M. Two concepts of accountability: Accountability as a virtue and as a mechanism. West European Politics, 33(5):946-967. 2010.

[24] Dubnick, M.J. Situating accountability: Seeking salvation for the core concept of modern governance. TR, University of New Hampshire. 2007.

[25] White, S.A., Business process modeling notation v1.0. Technical report. 2004.

[26] Katranuschkov, P., Gehre, A., Scherer, R.J., Reusable process patterns for collaborative work environments in AEC. 13th ICE, Nottingham, UK. 2007.

[27] Fox, J.A. The uncertain relationship between transparency and accountability. Development in Practice, 17(4):663-671. 2007.

[28] Karp, A.H., Haury, H., Davis, M.H. From ABAC to ZBAC: The Evolution of Access Control Models, Information Systems Security Assoc. J., 2010.

[29] Pete A. Epstein. Engineering of role/permission assignments. PhD thesis, Fairfax, VA, USA, 2002.

[30] H Roeckle, G. Schimpf, and R. Weidinger. Process-oriented approach for role-finding to implement role-based security administration in a large industrial organization. In RBAC '00: 5th ACM WS on RBAC, 2000.

[31] R. Crook, D. Ince, and B. Nuseibeh. Modelling access policies using roles in requirements engineering. Information and Software Technology, 45(14):979-991, 2003.

[32] R. Crook, D. Ince, and B.Nuseibeh. On modelling access policies: Relating roles to their organisational context. RE'05, pp 157-166, 2005.

[33] R. Chandramouli. A framework for multiple authorization types in a healthcare application system. 17th Annual Computer Security Applications Conference, ACSAC '01, Washington, DC, USA, 2001.

[34] E. B. Fernandez and J. C. Hawkins. Determining role rights from use cases. Second ACM WS on Role-based access control, NY, USA, 1997.

[35] G.Neumann and M. Strembeck. A scenario-driven role engineering process for functional rbac roles. SACMAT '02, New York, NY, USA, 2002

[36] A. Kern, M. Kuhlmann, A. Schaad, and J. Moffett. Observations on the role life-cycle in the context of enterprise security management. SACMAT '02, New York, NY, USA, 2002.

[37] J. Vaidya, V. Atluri, and Q. Guo. The role mining problem: Finding a minimal descriptive set of roles. SACMAT '07, NY, USA, 2007.